What is claimed is:

1. A method for co-verifying hardware and software, by using a host CPU, for a semiconductor device on which at least one target CPU and one OS are mounted, the hardware/software co-verification method comprising the steps of:

(a) inputting, as a verification model, a timed software component described in a C-based language and compiling the same, inputting as a verification model a hardware component described in the C-based language and compiling the same, and linking together the compiled timed software component and the compiled hardware component;

(b) inputting a testbench and compiling the same;

(c) linking together the verification models processed in step (a) and the testbench processed in step (b);

(d) performing a simulation based on an executing program generated in step (c); and

(e) outputting a result of the simulation performed in step (d).

2. A method for co-verifying hardware and software, by using a host CPU, for a semiconductor device on which at least one target CPU and one OS are mounted, the hardware/software co-verification method comprising the steps of:

(a) inputting, as a verification model, a timed software component constructed from binary code native to the host CPU, inputting as a verification model a hardware component described in a C-based language and compiling the same, and linking together the input timed software component and the compiled hardware component;

(b) inputting a testbench and compiling the same;

(c) linking together the verification models processed in step (a) and the testbench processed in step

(b);

(d) performing a simulation based on an executing program generated in step (c); and

(e) outputting a result of the simulation performed in step (d).

3. A method for co-verifying hardware and software, by using a host CPU, for a semiconductor device on which at least one target CPU and one OS are mounted, the hardware/software co-verification method comprising the steps of:

(a) inputting as a verification model a timed software component described in a C-based language and compiling the same, inputting as a verification model a timed software component constructed from binary code native to the host CPU and compiling the same, inputting as a verification model a hardware component described in the C-based language and compiling the same, and linking together the compiled or input timed software components and the compiled hardware component;

(b) inputting a testbench and compiling the same;

(c) linking together the verification models processed in step (a) and the testbench processed in step (b);

(d) performing a simulation based on an executing program generated in step (c); and

(e) outputting a result of the simulation performed in step (d).

4. The hardware/software co-verification method as claimed in claim 1 or 3 wherein, in order to generate in advance the timed software component described in the C-based language from an untimed software component described in ANSI-C, the method further comprises the steps of:

inputting the untimed software component described in ANSI-C, and recognizing basic blocks and inserting control points;

generating binary code native to a target CPU by compiling the untimed software component in which the control points have been inserted;

computing execution time between the control points in the generated binary code native to the target CPU; and

inserting, in accordance with the computed execution time, an execution time insertion statement at each of the control points inserted in the untimed software component, and thus outputting the timed software component described in the C-based language.

5. The hardware/software co-verification method as claimed in claim 2 or 3 wherein, in order to generate in advance the timed software component constructed from the binary code native to the host CPU from an untimed software component constructed from binary code native to a target CPU, the method further comprises the steps of:

inputting the untimed software component constructed from the binary code native to the target CPU, and converting the same into a software component expressed in the binary code native to the host CPU;

recognizing basic blocks and inserting control points in the software component expressed in the binary code native to the host CPU;

computing execution time between the control points in the software component in which the control points have been inserted; and

inserting, in accordance with the computed execution time, binary code functionally equivalent to an execution time insertion statement at each of the control points inserted in the software component, and thus outputting the timed software component constructed from the binary code native to the host CPU.